Activity: **5.1**
Determine Software Structure

Responsibility: Project Team Analysts

Description: A hierarchical approach is useful for determining the structure and components of the software product. Software system decomposition is one hierarchical approach that divides the software system into different levels of abstraction. Decomposition is an iterative process that continues until single purpose components (i.e., design entities or objects) can be identified. Decomposition is used to understand how the software product will be structured, and the purpose and function of each entity or object.

The goal of the decomposition is to create a highly cohesive, loosely coupled, and readily adapted design. A design exhibits a high degree of cohesion if each design entity in the program unit is essential for that unit to achieve its purpose. A loosely coupled design is composed of program units that are independent or almost independent.

Several reliable methods exist for performing system decomposition. Select a method that enables the design of simple, independent entities. Functional and object-oriented design are two common approaches to decomposition. These approaches are not mutually exclusive. Each may be applicable at different times in the design process.

The software system decomposition activity includes the following tasks.

5.1.1 Identify Design Entities
5.1.2 Identify Design Dependencies

Task: **5.1.1**
Identify Design Entities

Description: Design entities result from a decomposition of the software product requirements. A design entity is an element (or object) of a design that is structurally and functionally distinct from other elements and is separately named and referenced. The number and type of entities required to partition a design are dependent on a number of factors, such as the complexity of the software product, the design method used, and the programming environment. The objective of design entities is to divide the software product into separate components that can be coded, implemented, changed, and tested with minimal effect on other entities.

Attributes: A design entity attribute is a characteristic or property of a design entity. It provides a statement of fact about an entity. The following are common attributes that should be considered for each design entity.

- Assign a unique name to each entity.
- Classify each entity into a specific type. The type may describe the nature of the entity, such as a subprogram or module; or a class of entities dealing with a particular type of information.
- Describe the purpose or rationale for each entity. Include the specific functional and performance requirements for which the entity was created.
- Describe the function to be performed by each entity. Include the transformation applied to inputs by the entity to produce the desired output.
- Identify all of the external resources that are needed by an entity to perform its function.
- Specify the processing rules each entity will follow to achieve its function. Include the algorithm used by the entity to perform a specific task and contingency actions in case expected processing events do not occur.
- Describe the data elements internal to each entity. Include information such as the method of representation, format, and the initial and acceptable values of internal data. This description may be provided in the data dictionary.

Work Product: Maintain a record of all design entities. The records will be integrated into the Functional Design Document. Place a copy of the design entity information in the Project File.

Review Process: Schedule structured walkthroughs to verify that the design entities are correct, complete, and possess the required attributes.

Task: **5.1.2**
Identify Design Dependencies

Description: Design dependencies describe the relationships or interactions between design entities at the module, process, and data levels. These interactions may involve the initiation, order of execution, data sharing, creation, duplication, use, storage, or destruction of entities.

Identify the dependent entities of the software system design, describe their coupling, and identify the resources required for the entities to perform their function. Also define the strategies for interactions among design entities and provide the information needed to perceive how, why, where, and at what level actions occur.

Dependency descriptions should provide an overall picture of how the software product will work. Data flow diagrams, structure charts, and transaction diagrams are useful for showing the relationship among design entities.

The dependency descriptions may be useful in producing the system integration plan by identifying the entities that are needed by other entities and that must be developed first. Dependency descriptions can also be used to aid in the production of integration test cases.

Work Product: Add specific dependency information to the design entity records. The records will be integrated into the Functional Design Document. Place a copy of the dependency information in the Project File.

Review Process: Schedule structured walkthroughs to verify that the design dependencies are correct and complete.